KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Business and Economics

Models for the optimization of promotion campaigns: exact and heuristic algorithms

Fabrice T. Nobibon, Roel Leus and Frits C.R. Spieksma

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 0814

Models for the optimization of promotion campaigns: exact and heuristic algorithms

Fabrice T. Nobibon*, Roel Leus*, Frits C.R. Spieksma*
{Fabrice.TallaNobibon; Roel.Leus; Frits.Spieksma}@econ.kuleuven.be

Abstract. This paper presents an optimization model for the selection of sets of clients that will receive an offer for one or more products during a promotion campaign. The complexity of the problem makes it very difficult to produce optimal solutions using standard optimization methods. We propose an alternative set covering formulation and develop a branch-and-price algorithm to solve it. We also describe five heuristics to approximate an optimal solution. Two of these heuristics are algorithms based on restricted versions of the basic formulation, the third is a successive exact k-item knapsack procedure. A heuristic inspired by the Next-Product-To-Buy model and a depth-first branch-and-price heuristic are also presented. Finally, we perform extensive computational experiments for the two formulations as well as for the five heuristics.

Keywords: promotion campaign; minimum quantity commitment; integer programming; branch-and-price algorithm; non-approximability; heuristics; business-to-business; business-to-consumer.

1. Introduction

Promotion campaigns are fundamental direct marketing tools for improving the economic profit of a firm, either by acquiring new customers or by generating additional revenue from existing customers [13]. The former action is called "acquisition" while the latter is "retention" [27]. In this paper we are concerned with the latter case: campaigns that generate additional revenue by offering new products to existing customers. This study is justified by at least two practical facts. Reinartz et al. [27] point out that "When firms trade off between expenditures for acquisition and those for retention, a suboptimal allocation of retention expenditures will have a greater impact on long-term customer profitability than will suboptimal acquisition expenditures". Moreover, models and methods used for data analysis are more suited for retention [12] since more information is available. Retention boosts the customer lifetime value, which is defined by Kumar et al. [14] to be "the sum of cumulated cash flows – discounted using the weighted average cost of capital – of a customer over his or her entire lifetime with the firm". Customer lifetime value usually serves as a metric for a ranking or segmentation of the firm's customers [28]. During the last decades, the advances in data analysis coupled with the availability of customer data have pushed firms to develop a more customer-oriented strategy. Nowadays, such a strategy is globally accepted, but its practical implementation is far from being accomplished. This implementation delay is observed both in business-to-business and business-to-consumer settings, and is particularly pronounced in financial institutions such as large banks and insurance companies, which

*Operations Research Group, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000 Leuven, Belgium.

often have a large number of customers with full data available but may lack sophisticated tools that efficiently take into account these advantages in decision making [7].

In literature, promotion campaign models are also frequently referred to as optimal product targeting models [12]. The latter examine "Which products should be targeted to which customers to maximize profits, under the constraints that only a limited number can be targeted to each customer, and each product has a minimum sales target", which is mentioned by [12] as an interesting issue for future research. A promotion campaign problem is essentially characterized by two steps, which are "data analysis" and "problem formulation and solution". The first step, which is mainly statistical, has received an increasing attention with the advances in data analysis. Recently, numerous models that carry the name "response models" have been developed and are currently being used in practice [5, 12]. Although this step is necessary for an application in financial institutions (as its outputs are used as inputs for the second step), its use can be less important for an application in other areas. De Reyck and Degraeve [6], for example, have developed a model mainly based on the second step for an advertisement company.

This paper investigates the development of optimization models for promotion campaigns based on integer programming. Motivation for studying this problem comes from a case occurring at FORTIS [10], one of the leading banks in Belgium. We aim to maximize the profit (return on investment) subject to business constraints such as the campaign's return on investment hurdle rate that must be met, a limitation on the funding available for each product, a restriction on the maximum number of possible offers to a client and a minimum quantity commitment (MQC) on the number of units of product to be offered in order for that product to be part of the campaign. This constraint has been briefly mentioned by Cohen [5] as technical issue for an application in a bank. However, he did not explicitly incorporate it into his model. Our model takes into account this constraint, making it an extension of the model used by Cohen. In our formulation, we also impose a more general version of the MQC constraint, allowing the fixed minimum quantity to depend on the product, which distinguishes the constraint from comparable MQC restrictions studied in the analysis of transportation problems [18], bottleneck problems [19], and assignment problems [17].

In this paper, we present a basic integer programming formulation for the optimization of promotion campaigns. We show the non-approximability of the problem, which makes the existence of an algorithm that will always provide a feasible solution and guarantee a specified proportion of optimal profit in polynomial time, highly unlikely. We next present a set covering formulation and develop a branch-and-price algorithm for solving it. A dynamic programming algorithm and a 2-approximation algorithm are presented for solving the pricing problem, which is closely related to the k-item knapsack problem. The size of instances that can be solved optimally using this algorithm allows its efficient use for business-to-business promotion campaigns (which have moderate size and high variable and fixed costs) and for sampling approaches in financial institutions [5]. We then present five heuristics to approximate an optimal solution, which can be used for large instances and hence for business-to-consumer promotion campaigns. These heuristics are either variants of the algorithms used in practice for application in a bank (see [10]) or developed based on the structure of the problem.

This paper is organized as follows. Section 2 describes the basic integer programming formulation for deciding on the composition of promotion campaigns. The complexity of

the basic formulation makes it very difficult to produce optimal solutions using standard optimization methods. We propose an alternative formulation called the set covering formulation in Section 3 and develop a branch-and-price algorithm to solve it. In Section 4, we describe five heuristics to approximate an optimal solution. The first two are algorithms based on restricted versions of the basic formulation, while the third is a successive exact k-item knapsack procedure (E-kKP), the fourth is a heuristic inspired by the Next-Product-To-Buy model used by Knott et al. [12] and the last one is a depth-first branch-and-price heuristic. Section 5 contains some ideas pertaining to the implementation issue of these algorithms. The experimental results for the two formulations (basic formulation and set covering formulation) as well as for the five heuristics are presented in Section 6. Finally, two extensions of the studied model are presented in Section 7, followed by some conclusions in Section 8.

2. Basic formulation

The objective of a promotion campaign is to find a way to achieve a maximum profit by offering n different products to m customers while taking into account various business constraints. We incorporate the following restrictions: the return on investment hurdle rate must be met for the campaign, the budget allocated to each product is limited, an upper bound is imposed on the number of products that can be offered to each client and there is also a MQC constraint for each product. We define the parameter r_{ij} as the probability that client i reacts positively to an offer of product j (or the probability that product j is the next product bought by client i [12]) and DFV_{ij} as the expected return for the firm when client i responds positively to the offer of product j. The latter is termed the Delta Financial Value by FORTIS [10]. These two parameters are the basis for the computation of customer lifetime value [28]. Practically, these parameters are estimated using response models based on historical data [5, 12, 26] and are assumed to be available within the firm. Further, there is a variable cost c_{ij} associated with the offer of product j to client i, the upper bound M_i of offers that can be made to a client i (this quantity is related to the status of the client), the minimum quantity commitment bound O_j associated with product j, the budget B_j allocated to the product j, a fixed cost f_j needed for a product j if used for the campaign and finally the corporate hurdle rate R. The value of R is dependent on the firm and the riskiness of the investment. In practice, most firms use their weighted average cost of capital (WACC) as an estimation of R [3]. Let x_{ij} and y_j be binary decision variables defined by:

$$x_{ij} = \begin{cases} 1, & \text{if product } j \text{ is offered to client } i, \\ 0, & \text{otherwise,} \end{cases}$$

$$y_j = \begin{cases} 1, & \text{if product } j \text{ is used during the campaign,} \\ 0, & \text{otherwise.} \end{cases}$$

A basic formulation for the promotion campaign problem can be expressed as:

$$\text{(M1)} \quad \text{maximize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} \left(r_{ij} DFV_{ij} - c_{ij}\right) x_{ij} - \sum_{j=1}^{n} f_j y_j \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij} DFV_{ij} x_{ij} \geq (1+R)\left[\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j y_j\right] \tag{2}$$

$$\sum_{i=1}^{m} c_{ij} x_{ij} \leq B_j \qquad j = 1, \ldots, n, \tag{3}$$

$$\sum_{j=1}^{n} x_{ij} \leq M_i \qquad i = 1, \ldots, m, \tag{4}$$

$$\sum_{i=1}^{m} x_{ij} \leq m y_j \qquad j = 1, \ldots, n, \tag{5}$$

$$\sum_{i=1}^{m} x_{ij} \geq O_j y_j \qquad j = 1, \ldots, n, \tag{6}$$

$$y_j, x_{ij} \in \{0,1\} \qquad i = 1 \ldots, m,\ j = 1, \ldots, n. \tag{7}$$

The objective function (1) is the maximization of the total net benefit received from the offer of products to clients minus the fixed cost of using the products for the campaign. The first constraint (2) is the corporate hurdle rate constraint, which makes sure that the campaign's return on investment is at least R, and was first suggested by Cohen [5] for an application in a bank. The set of constraints (3) enforces that we should not exceed the budget B_j allocated to the product j. Here, the product dependency of the budget reflects the situation in a large firms where an individual business unit is responsible for the production and the sale of a product. Hence, each business unit has its own budget. The set of constraints (4) states that we cannot propose more than a certain number M_i of products to client i; the sets of constraints (5) and (6) constitute the MQC constraint, which specifies that a product j taking part in the campaign will be offered to at least a certain number O_j of clients ($O_j > 0$), and finally the last set of constraints (7) is the integrality constraint. We denote by p_{ij} the return to the firm (revenue) when client i reacts positively to the offer of product j, so $p_{ij} = r_{ij} DFV_{ij}$. In the remainder of this paper, we will mostly use p_{ij}.

The graph depicted in Figure 1 represents an instance of the promotion campaign problem with two products and three clients.

Example 1. *An integer program corresponding to the example described by Figure 1 is given by:*

$$\begin{aligned}
\textit{maximize} \quad & -2x_{11} + 3x_{21} + 3x_{31} + x_{12} - 2x_{22} + 2x_{32} \\
\textit{subject to} \quad & -\frac{8}{3}x_{11} + \frac{8}{3}x_{21} + \frac{5}{3}x_{31} - \frac{1}{3}x_{12} - \frac{8}{3}x_{22} + \frac{4}{3}x_{32} \geq 0 \\
& 2x_{11} + x_{21} + 4x_{31} \leq 4, \quad 4x_{12} + 2x_{22} + 2x_{32} \leq 5 \\
& x_{11} + x_{12} \leq 1, \quad x_{21} + x_{22} \leq 2, \quad x_{31} + x_{32} \leq 1 \\
& x_{11} + x_{21} + x_{31} \geq 2y_1, \quad x_{12} + x_{22} + x_{32} \geq 2y_2 \\
& x_{11} + x_{21} + x_{31} \leq 3y_1, \quad x_{12} + x_{22} + x_{32} \leq 3y_2 \\
& y_j, x_{ij} \in \{0,1\} \qquad i = 1,2,3, \quad j = 1,2.
\end{aligned}$$

An optimal solution offers Product 1 *to client* 1 *and client* 2*. So, we have* $x_{11} = x_{21} = y_1 = 1$, $x_{31} = 0$ *and* $x_{12} = x_{22} = x_{32} = y_2 = 0$ *for an optimal objective value of* 1*. On the*

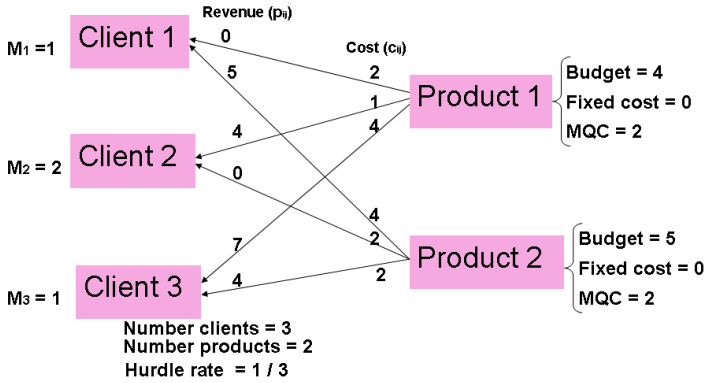

Figure 1: An example of a promotion campaign with two products and three clients.

other hand, an optimal solution to the linear programming relaxation of this example is $x_{21} = x_{31} = \frac{4}{5}$, $x_{12} = x_{32} = \frac{1}{5}$, $y_1 = \frac{4}{5}$, $y_2 = \frac{1}{5}$ *with optimal objective value* 5.4.

Definition 1. *A* **non-trivial feasible solution** *for the basic formulation (M1) is a feasible solution which achieves a non-zero objective value.*

The following result shows that there is little hope for finding a polynomial time algorithm for solving (M1).

Proposition 1. *The promotion campaign problem defined by the formulation (M1) is strongly NP-hard, even for* $O_j = 1$ *for all* j.

Proof: This result follows directly from the fact that the basic formulation (M1) has the generalized assignment problem (GAP) [21, 29] as a special case for $R = 0$, $M_i = 1$, $\forall i$, and $O_j = 1$, $f_j = 0$, $\forall j$ and $p_{ij} \geq c_{ij}$ for all i and j. The latter problem is known to be strongly NP-hard [29]. □

Moreover, the basic formulation (M1) is difficult to solve even approximately. We prove this non-approximability result by showing that it is NP-hard to find a non-trivial feasible solution for (M1). The proof uses the following variant of Partition (see [8]; by adding $|A|$ dummy elements of size 0 to an instance of the usual Partition problem, one easily sees that this variant of Partition is as hard as the original problem):
INSTANCE: A finite set $A = \{1, 2, \ldots, 2q\}$ with size $s(i) \in \mathbb{Z}$ for each $i \in A$, $K = \frac{1}{2}\sum_{i \in A} s(i)$.
QUESTION: Does there exist a subset $A' \subset A$ with $|A'| = q$ and $\sum_{i \in A'} s(i) = K$?

Proposition 2. *Finding a non-trivial feasible solution for the basic formulation (M1) is NP-hard.*

Proof: For a given arbitrary instance of Partition, consider the following polynomial reduction to an instance of (M1). Each $i \in A$ indicates a client with a unit upper bound, so $m = 2q$ and $M_i = 1$ for all $i = 1, 2, \ldots, 2q$. Suppose there is one product, $n = 1$. Let $\delta = 2K + 1$. For each client $i = 1, 2, \ldots, 2q$, the cost $c_{i1} = \delta + s(i)$, the revenue $p_{i1} = \delta s(i)$. Suppose also that for our product, product 1, the lower bound $O_1 = q$, the budget $B_1 = q\delta + K$, and the fixed cost $f_1 = 0$. Finally, we set the hurdle rate $R = \frac{(K-q)\delta - K}{q\delta + K}$. Now we prove that a non-trivial feasible solution for (M1) exists if and only if there is a solution A' to Partition.

On one hand, if Partition has a solution A', we offer the product to the clients in A'. This is a non-trivial feasible solution to the instance for (M1) constructed above since: (i) the lower bound $O_1 = q$ is met, (ii) the budget is met ($q\delta + \sum_{i \in A'} s(i) = q\delta + K = B_1$) and (iii) the hurdle rate is achieved: $\sum_{i \in A'} p_{i1} = \delta \sum_{i \in A'} s(i) = \delta K = \frac{\delta K}{q\delta + K}(q\delta + K) = \left(1 + \frac{(K-q)\delta - K}{q\delta + K}\right)(q\delta + K) = (1 + R)(q\delta + K)$.

On the other hand, if we have a non-trivial feasible solution, then consider the set of clients A' receiving the product. We have $|A'| = q$ and $\sum_{i \in A'} c_{i1} = \sum_{i \in A'} \delta + s_i \leq B_1 = q\delta + K$. In fact, we have $\sum_{i \in A'} c_{i1} = q\delta + K$. Suppose that $\sum_{i \in A'} c_{i1} < q\delta + K$, then $\sum_{i \in A'} c_{i1} = q\delta + K_1$ with $K_1 < K$. Thus

$$\begin{aligned} \sum_{i \in A'} p_{i1} &= K_1\delta = K\delta\frac{K_1}{K} < K\delta\frac{q\delta + K_1}{q\delta + K} = (q\delta + K_1)\frac{K\delta}{q\delta + K} \\ &= \left(1 + \frac{(K-q)\delta - K}{q\delta + K}\right)(q\delta + K_1) = (1 + R)(q\delta + K_1), \end{aligned}$$

contradicting the fact that we have a non-trivial feasible solution because the corporate hurdle rate constraint is not satisfied. Therefore $|A'| = q$ and $\sum_{i \in A'} c_{i1} = q\delta + K$, which implies that A' is a solution to the variant of Partition defined above. □

The results of Proposition 1 and Proposition 2 justify the intensive use of heuristics in practice [5, 6, 12].

The basic formulation (M1) can be strengthened by using the following disaggregate version of constraints (5):

$$x_{ij} \leq y_j \quad i = 1, \ldots, m,\ j = 1, \ldots, n. \tag{8}$$

The following result allows the relaxation of the integrality constraint on y_j for all j.

Proposition 3. *The convex hull of the feasible solutions to the integer program (M1) is identical to the convex hull of the solutions that satisfy the constraints (2), (3), (4), (6), (8) and* $0 \leq y_j \leq 1$, $x_{ij} \in \{0, 1\}$ *for all* i, j.

Proof: Denote the convex hull of the feasible solutions to (M1) by CH(M1) and the convex hull of the solutions that satisfy the constraints (2), (3), (4), (6), (8) and $0 \leq y_j \leq 1$, $x_{ij} \in \{0, 1\}$ for all i, j by CH^*. Because any feasible solution to (M1) satisfies (2), (3), (4), (6), (8) and $0 \leq y_j \leq 1$, $x_{ij} \in \{0, 1\}$, the set of feasible solutions to (M1) is included in CH^*, therefore, the convexity of CH^* implies that $CH(\text{M1}) \subseteq CH^*$.

On the other hand, let (x^0, y^0) be an extreme point of CH^*, (x^0, y^0) satisfies (2), (3), (4), (6) and $x_{ij} \in \{0, 1\}$, and (x^0, y^0) satisfies (8) which implies that (x^0, y^0) satisfies (5).

Furthermore $y_j^0 \in \{0,1\}$ for all j because if there was a j_0 such that $0 < y_{j_0}^0 < 1$, then $x_{ij_0}^0 = 0$ for all i and constraints (6) will be violated. We then have $y_j^0 \in \{0,1\}$ for all j and therefore (x^0, y^0) is a feasible solution to (M1), so belongs to $CH(\text{M1})$. Since (x^0, y^0) is an extreme point of CH^*, it is also an extreme point of $CH(\text{M1})$ (because $CH(\text{M1}) \subseteq CH^*$) implying that $CH^* \subseteq CH(\text{M1})$. □

A similar proof has been derived by Lim et al. in [18] for the transportation problem with minimum quantity commitment.

3. Branch-and-price

This section is devoted to the application of a branch-and-price algorithm for solving the promotion campaigns problem. In the first subsection, we derive an appropriate formulation, called a set covering formulation. The next subsection studies the pricing problem and the last subsection presents a branching strategy.

3.1 A set covering formulation

Proposition 3 and the inequalities (8) lead to the following strengthened formulation of our problem:

$$
\begin{aligned}
(\text{M2}) \quad \textsf{maximize} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} (p_{ij} - c_{ij})\, x_{ij} - \sum_{j=1}^{n} f_j y_j && && (9)\\
\textsf{subject to} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij} x_{ij} \geq (1+R)\left[\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{j=1}^{n} f_j y_j\right] && && (10)\\
& \sum_{j=1}^{n} x_{ij} \leq M_i && i = 1,\ldots,m, && (11)\\
& \sum_{i=1}^{m} c_{ij} x_{ij} \leq B_j && j = 1,\ldots,n, && (12)\\
& \sum_{i=1}^{m} x_{ij} \geq O_j y_j && j = 1,\ldots,n, && (13)\\
& x_{ij} \leq y_j \leq 1 && i = 1,\ldots,m,\ j = 1,\ldots,n, && (14)\\
& y_j \geq 0,\ x_{ij} \in \{0,1\} && i = 1,\ldots,m,\ j = 1,\ldots,n. && (15)
\end{aligned}
$$

We investigate a relaxation of (M2), in which we consider (10) and (11) to be *coupling constraints* and for each product $j = 1,\ldots,n$, we have the *special constraints* given by (12), (13), (14) and (15). For $j = 1,\ldots,n$, we define $A_j = \{(x_{.j}, y_j);\ \sum_{i=1}^{m} c_{ij} x_{ij} \leq B_j,\ \sum_{i=1}^{m} x_{ij} \geq O_j y_j,\ x_{ij} \leq y_j,\ i = 1,\ldots,m,\ y_j \leq 1,\ y_j \geq 0,\ x_{.j} \in \{0,1\}^m\}$, where $x_{.j}$ is a vector with m entries x_{ij}. Because $(0_{.j}, 0) \in A_j$, A_j is nonempty. Furthermore, A_j is bounded as $0 \leq y_j, x_{ij} \leq 1$. For every product j, $A_j = \{(x_{.j}^0, y_j^0), (x_{.j}^1, y_j^1), (x_{.j}^2, y_j^2), \ldots, (x_{.j}^{k_j}, y_j^{k_j})\}$. Explicitly, A_j consists of $(0_{.j}, 0)$ and all the $(x_{.j}^0, 1)$ with $\sum_{i=1}^{m} c_{ij} x_{ij} \leq B_j$, $\sum_{i=1}^{m} x_{ij} \geq O_j$, $x_{ij} \in \{0,1\}$,

that is the empty set of clients coupled with 0 and any subset of clients of cardinality greater than or equal to O_j and total cost less than or equal to B_j, coupled with 1. We assume that $(x^0_{\cdot j}, y^0_j) = (0_{\cdot j}, 0)$ so that $A_j = \{(0_{\cdot j}, 0), (x^1_{\cdot j}, 1), (x^2_{\cdot j}, 1), \ldots, (x^{k_j}_{\cdot j}, 1)\}$.

We relax (M2) by considering $conv(A_j)$ for each product j, where $conv(A_j)$ is the convex hull of A_j. Any element $(x_{\cdot j}, y_j) \in conv(A_j)$ is a convex combination of its extreme vertices and hence can be represented in the form $(x_{\cdot j}, y_j) = \sum_{p=0}^{k_j} z_{pj}(x^p_{\cdot j}, y^p_j)$ where the coefficients z_{pj} are nonnegative and satisfy $\sum_{p=0}^{k_j} z_{pj} = 1$, $j = 1, \ldots, n$. A Dantzig-Wolfe decomposition of the relaxation of (M2) is then given by the master problem (MP).

$$\text{(MP)} \quad \text{maximize} \quad \sum_{j=1}^{n} \left[\sum_{p=0}^{k_j} \left[\left(\sum_{i=1}^{m} (p_{ij} - c_{ij}) x^p_{ij} \right) - f_j y^p_j \right] z_{pj} \right] \tag{16}$$

$$\text{subject to} \quad \sum_{j=1}^{n} \left[\sum_{p=0}^{k_j} \left[\left(\sum_{i=1}^{m} (p_{ij} - (1+R)c_{ij}) x^p_{ij} \right) - (1+R) f_j y^p_j \right] z_{pj} \right] \geq 0 \tag{17}$$

$$\sum_{j=1}^{n} \sum_{p=0}^{k_j} x^p_{ij} z_{pj} \leq M_i \qquad i = 1, \ldots, m, \tag{18}$$

$$\sum_{p=0}^{k_j} z_{pj} = 1 \qquad j = 1, \ldots, n, \tag{19}$$

$$z_{pj} \geq 0 \qquad j = 1, \ldots, n,\ p \in \{0, 1, \ldots, k_j\}. \tag{20}$$

In what follows, we will consider the subset of extreme points without $(0_{\cdot j}, 0)$, subsequently denoted as $\bar{A}_j = \{(x^1_{\cdot j}, 1), (x^2_{\cdot j}, 1), \ldots, (x^{k_j}_{\cdot j}, 1)\}$. This change requires that each constraint in (19) become an inequality in order for the optimal value to remain unaffected.

For every product j, we map each extreme point $(x^p_{\cdot j}, y^p_j) \in \bar{A}_j$ to S_{pj} where $S_{pj} = \{i \in \{1, \ldots, m\} \mid x^p_{ij} = 1\}$. Using the equalities $\sum_{i=1}^{m} p_{ij} x^p_{ij} = \sum_{i \in S_{pj}} p_{ij}$ and $\sum_{i=1}^{m} c_{ij} x^p_{ij} = \sum_{i \in S_{pj}} c_{ij}$, an integer programming version of (MP) is

$$\text{(M3)} \quad \text{maximize} \quad \sum_{j=1}^{n} \left[\sum_{p=1}^{k_j} \left(\sum_{i \in S_{pj}} (p_{ij} - c_{ij}) - f_j \right) z_{pj} \right] \tag{21}$$

$$\text{subject to} \quad \sum_{j=1}^{n} \left[\sum_{p=1}^{k_j} \left(\sum_{i \in S_{pj}} (p_{ij} - (1+R)c_{ij}) - (1+R) f_j \right) z_{pj} \right] \geq 0 \tag{22}$$

$$\sum_{j=1}^{n} \sum_{p : i \in S_{pj}} z_{pj} \leq M_i \qquad i = 1, \ldots, m, \tag{23}$$

$$\sum_{p=1}^{k_j} z_{pj} \leq 1 \qquad j = 1, \ldots, n, \tag{24}$$

$$z_{pj} \in \{0, 1\} \qquad j = 1, \ldots, n,\ p \in \{1, \ldots, k_j\}. \tag{25}$$

In the formulation (M3), the binary variable z_{pj} indicates whether the product j is offered to the set of clients S_{pj} ($z_{pj} = 1$) or not ($z_{pj} = 0$). The first constraint (22) enforces that

the campaign's return on investment must be at least R, the set of constraints (23) ensures that at most M_i products are offered to client i and the set of constraints (24) states that at most one nonempty set of clients is selected for each product. We call the formulation (M3) a *set covering formulation* to reflect the fact that its solution can be viewed as a cover for the set of clients. The set covering formulation (M3) is then essentially obtained by applying Dantzig-Wolfe decomposition to a relaxation of the basic formulation (M2). As a consequence, the value of the bound provided by the LP relaxation of (M3) is equal to the value of the Lagrangian dual obtained by dualizing the constraints (2) and (4) [24]. Furthermore, the example described by Figure 1 illustrates that the LP relaxation of (M3) is stronger than the LP relaxation of (M1). We denote $S_{11} = \{1, 2\}$, $S_{12} = \{2, 3\}$ and z_{11}, z_{12} the corresponding variables. The associated LP relaxation of (M3) is

$$\begin{array}{ll} \text{maximize} & 1z_{11} + 0z_{12} \\ \text{subject to} & 0z_{11} - \dfrac{4}{3}z_{12} \geq 0 \\ & 0 \leq z_{1j} \leq 1 \qquad\qquad j = 1, 2. \end{array}$$

Remark that the constraints (23) and (24) are obviously satisfied. The optimal solution is $(1, 0)$ with an optimal value of 1, which is also the optimal objective function value of the integer program. The result highlighted in the above example has been observed for the GAP by Savelsbergh [29].

Let z be any feasible solution to the LP relaxation of (M3) and let $x^*_{ij} = \sum_{p:i\in S_{pj}} z_{pj}$, $y^*_j = \sum_{p=1}^{k_j} z_{pj}$, then (x^*, y^*) is a feasible solution to the LP relaxation (LPM2) of (M2). Furthermore, we have the following result, which is useful for the branching strategies.

Proposition 4. *Given a feasible solution z to the LP relaxation of (M3), for a given product j, if z_{pj} is fractional, then there must be an i such that $x^*_{ij} = \sum_{p:i\in S_{pj}} z_{pj}$ is fractional.*

Proof: Suppose there is no client i such that x^*_{ij} is fractional. Let $F = \{p \in \{1, \ldots, k_j\} \,|\, 0 < z_{pj} < 1\}$ be the set of fractional variables associated with product j. We may assume that $|F| \geq 2$, otherwise the hypothesis (there is no client i such that x^*_{ij} is fractional) will be violated. We have $\sum_{p\in F} z_{pj} = y^*_j \leq 1$ for $j = 1, \ldots, n$ and $\sum_{p\in F:i\in S_{pj}} z_{pj} = x^*_{ij} \in \{0, 1\}$, $i = 1, \ldots, n$. In particular for $i \in \cup_{p\in F} S_{pj}$, we have $\sum_{p\in F:i\in S_{pj}} z_{pj} = 1$. But $\sum_{p\in F:i\in S_{pj}} z_{pj} = \sum_{p\in F} x^p_{ij} z_{pj} = 1$ and $\sum_{p\in F} z_{pj} = 1$. Hence, $x^p_{ij} = 1$, $\forall p \in F$, meaning that for all $p_1, p_2 \in F$, we have $S_{p_1} = S_{p_2}$, contradicting $|F| \geq 2$. □

Proposition 4 implies that the bound provided by the LP relaxation of the set covering formulation is stronger than that obtained by the LP relaxation of the basic formulation.

3.2 The pricing problem

We consider the LP relaxation (LPM3) of (M3) obtained by replacing constraints (25) by

$$z_{pj} \geq 0 \qquad\qquad j = 1, \ldots, n,\ p \in \{1, \ldots, k_j\}. \tag{26}$$

Formulation (LPM3) has an exponential number of variables, making it difficult to solve even for average size instances. Instead of solving the full master problem (LPM3), we consider

a restricted problem that includes only a subset of variables (columns) and can be solved directly. Additional columns for the restricted problem can be generated by looking at the dual of (LPM3) given by:

$$\begin{aligned}
\text{(DLPM3)} \quad \textsf{minimize} \quad & \sum_{i=1}^{m} M_i u_i + \sum_{j=1}^{n} v_j \\
\textsf{subject to} \quad & \sum_{i \in S_{pj}} [(p_{ij} - (1+R)c_{ij})d + u_i] - (1+R)f_j d + v_j \\
& \geq \sum_{i \in S_{pj}} (p_{ij} - c_{ij}) - f_j, \; \forall p, j, \quad (27) \\
& u_i, v_j \geq 0, \; d \leq 0 \qquad i = 1, \ldots, m, \; j = 1, \ldots, n, \quad (28)
\end{aligned}$$

where we use the dual variables d corresponding to (22), u_i corresponding to the set of constraints (23) and v_j corresponding to (24). The optimal solution found for the restricted problem is not optimal for the master problem if the associated dual variables u, v and d violate one of the constraints (27). Note that the set of constraints (28) is automatically satisfied by the dual variables. Since it is not computationally viable to compute and check the inequality (27) for all couples (p, j) not included in the restricted problem, we propose to proceed as follows. First, we drop the index j and assume that the product is given. We then solve the following question called the *pricing problem*:

$$\exists S_p \text{ such that } \sum_{i \in S_p} [p_i(d-1) + c_i(1-(1+R)d) + u_i] + v + f(1-(1+R)d) < 0\,? \quad (29)$$

Remark that for a given product j, the left hand side of the inequality is exactly the reduced cost of the variable z_{pj}, so that the pricing problem (29) checks the primal optimality condition [2].

Solving the pricing problem

A solution to the pricing problem (29) for a fixed product j can be obtained by solving the following variant of the *k-item knapsack problem* (kKP). For ease of exposition, we define $w_i = p_i(d-1) + c_i(1-(1+R)d) + u_i$ for all i.

$$\begin{aligned}
\text{(kKP)} \quad \textsf{minimize} \quad & \sum_{i=1}^{m} w_i x_i & (30) \\
\textsf{subject to} \quad & \sum_{i=1}^{m} c_i x_i \leq B & (31) \\
& \sum_{i=1}^{m} x_i \geq O & (32) \\
& x_i \in \{0,1\} \qquad i = 1, \ldots, m. & (33)
\end{aligned}$$

We take the w_i's as the weights and the c_i's as the costs. Notice that in general the problem (kKP) is weakly NP-hard. Furthermore in our case, the w_i need not always be positive nor

are they always integers. These last observations make the use of dynamic programming by weight [11] for solving (kKP) inefficient. Kellerer et al. [11] derive a dynamic programming algorithm for the case where the cardinality constraint (32) is replaced by $\sum_{i=1}^{m} x_i \leq O$. We show here that this algorithm is easily modified to deal with (30)−(33).

Exact algorithm

We propose a dynamic programming algorithm based on the algorithm describe in [11] for solving (kKP) in pseudopolynomial time. Let k be the optimal value of the following problem

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} x_i \\ \text{subject to} \quad & \sum_{i=1}^{m} c_i x_i \leq B \\ & x_i \in \{0,1\} \qquad i = 1, \ldots, m. \end{aligned}$$

Clearly, since $c_i \geq 0$, k can be computed by taking the items with smallest cost c_i until the constraint is violated. If $k < O$ then the problem (kKP) is infeasible. Assuming that $k \geq O$, we define the two-dimensional dynamic programming function $Y_i(c,l)$ for $i = 0, 1, \ldots, m$; $l = 0, 1, \ldots, k$; $c = 0, 1, \ldots, B$, as the optimal solution value of the following problem:

$$Y_i(c,l) = \min \left\{ \sum_{j=1}^{i} w_j x_j \mid \sum_{j=1}^{i} c_j x_j = c, \sum_{j=1}^{i} x_j = l, x_j \in \{0,1\} \right\}$$

An entry $Y_i(c,l) = q$ means that among the clients $1, 2, \ldots, i$, there exists a subset of exactly l of these clients with total cost c and minimal weight q among all such subsets.

The initialization is given by $Y_0(c,l) = +\infty$ for $l = 0, 1, \ldots, k$; $c = 0, 1, \ldots, B$ and $Y_0(0,0) = 0$. Then, for $i = 1, \ldots, m$, the entries Y_i can be computed from those of Y_{i-1} by the following recursion

$$Y_i(c,l) = \begin{cases} Y_{i-1}(c,l) & \text{if } c_i > c \\ \min\{Y_{i-1}(c,l), Y_{i-1}(c-c_i, l-1) + w_i\} & \text{if } l > 0,\ c_i \leq c. \end{cases} \tag{34}$$

After computing all the values of Y from Y_1 up to Y_m, the optimal objective function value of (kKP) is given by

$$\min\{Y_m(c,l) \mid l \geq O,\ \ c = 0, 1, \ldots, B\}.$$

We observe that only entries in $Y_{i-1}(c,l)$ need to be stored in order to derive $Y_i(c,l)$. Caprara et al. [4] propose an implementation based on pointers that achieves a space complexity of $O(k^2 B)$ and a time complexity of $O(mkB)$.

Approximation algorithm

An approximation algorithm for (kKP) is based on the LP relaxation of the problem, which is obtained by replacing constraints (33) by

$$0 \leq x_i \leq 1 \quad i = 1, \ldots, m. \tag{35}$$

Denoting the optimal objective function value of the LP relaxation of (kKP) by z^{LP}, the following result proved in [4, 11] holds.

Lemma 1. *An optimal basic solution x^* of the LP relaxation (30), (31), (32) and (35), has at most two fractional components. Let $J_1 := \{l \,|\, x_l^* = 1\}$. If the basic solution has two fractional components x_i^* and x_j^*, supposing without loss of generality $c_j \leq c_i$, then $w_i + \sum_{l \in J_1} w_l \leq z^{LP}$ and the solution defined by $J_1 \cup \{j\}$ is feasible for the (kKP).*

The following algorithm describes an approximation algorithm for (kKP).

1 Approximation algorithm for (kKP)

Let x^{LP} be an optimal solution of the LP relaxation of (kKP)
$J_1 := \{l \,|\, x_l^{LP} = 1\}$
$F := \{l \,|\, 0 < x_l^{LP} < 1\}$ // fractional variables
if $F = \emptyset$ **then**
  $z^A := z^{LP}$
end if
if $F = \{i\}$ **then**
  $z^A := \min\{\sum_{l \in J_1} w_l, \sum_{l \in \{i\} \cup R_i} w_l\}$ where R_i is the set of the $k-1$ items with the smallest weight in $\{1, \ldots, m\} \setminus \{i\}$
end if
if $F = \{i, j\}$ with $c_j \leq c_i$ **then**
  $z^A := \min\{\sum_{l \in J_1 \cup \{j\}} w_l, \sum_{l \in \{i\} \cup R_i} w_l\}$ where R_i is the set of the $k-1$ items with the smallest weight in $\{1, \ldots, m\} \setminus \{i\}$
end if

Proposition 5. *Approximation algorithm for (kKP) is a 2-approximation algorithm for (kKP) and runs in $O(m)$ time.*

Proof: This follows from an easy modification of a proof in [4, 11]. □

Using the column generation procedure outlined above, we can solve the master problem (LPM3) in reasonable time. There is no guarantee, however, that the solution found will be an integer solution; if this is not the case we will proceed with a branch-and-bound algorithm.

3.3 Branch-and-bound

Branching is needed when the optimal solution to (LPM3) turns out not to be integral. However, as Savelsbergh points out in [29], a naive branching strategy may sometimes lead to a conflict between the variable used for branching and the one generated. Hence, it is worth looking for a branching strategy compatible with the pricing problem.

Proposition 4 allows the use of a hybrid branching policy [1, 29]; that is to perform branching using the basic formulation while working with the set covering formulation. We will then fix a single variable (variable dichotomy) [29,31]. This variable dichotomy branching strategy is exactly the branching scheme proposed by Ryan and Foster for set partitioning master problems [1].

In the basic formulation, fixing x_{ij} to zero forbids product j to be offered to client i, and fixing x_{ij} to one requires product j to be offered to client i. In the set covering formulation, this is done by adding (not explicitly) one extra constraint. Fixing x_{ij} to zero leads to $\sum_{p:i\in S_{pj}} z_{pj} = 0$ and fixing to one leads to $\sum_{p:i\in S_{pj}} z_{pj} = 1$. Hence, at the node u of the tree, let $H(u) \subseteq \{1,\ldots,n\}$ be the subset of products j for which there exists a non-empty set of clients $R_j^u \subseteq \{1,\ldots,m\}$, $(R_j^u \neq \emptyset)$ who must receive an offer of product j. Similarly, let $L(u) \subseteq \{1,\ldots,n\}$ be the set of products j for which there exists a non-empty set of clients $N_j^u \subseteq \{1,\ldots,m\}$, $(N_j^u \neq \emptyset)$ who cannot receive an offer of product j. The LP problem to be solved at the node u is the following:

$$(\mathrm{LP}_u) \quad \text{maximize} \quad \sum_{j=1}^{n}\left[\sum_{p=1}^{k_j}\left(\sum_{i\in S_{pj}}(p_{ij}-c_{ij})-f_j\right)z_{pj}\right] \tag{36}$$

$$\text{subject to} \quad \sum_{j=1}^{n}\left[\sum_{p=1}^{k_j}\left(\sum_{i\in S_{pj}}(p_{ij}-(1+R)c_{ij})-(1+R)f_j\right)z_{pj}\right] \geq 0 \tag{37}$$

$$\sum_{j=1}^{n}\sum_{p:i\in S_{pj}} z_{pj} \leq M_i \qquad i=1,\ldots,m, \tag{38}$$

$$\sum_{p:R_j^u\subseteq S_{pj}} z_{pj} = 1 \qquad j\in H(u), \tag{39}$$

$$\sum_{p:N_j^u\cap S_{pj}\neq\emptyset} z_{pj} = 0 \qquad j\in L(u), \tag{40}$$

$$\sum_{p=1}^{k_j} z_{pj} \leq 1 \qquad j=1,\ldots,n, \tag{41}$$

$$z_{pj} \geq 0 \qquad j=1,\ldots,n,\ p\in\{1,\ldots,k_j\}. \tag{42}$$

The branching scheme resulting from the variable dichotomy branching strategy is compatible with the pricing problem for it does not render the pricing problem more difficult. To meet the set of constraints (40), we set $x_i = 0$, $\forall i \in N_j^u$ when solving the pricing problem associated with the product $j \in L(u)$. Similarly, for the set of constraints (39), we set $x_i = 1$, $\forall i \in R_j^u$ when solving the pricing problem corresponding to the product $j \in H(u)$. The constraints (31) and (32) are updated and the remaining problem is still a (kKP), of reduced size. Moreover, the inequality (24) of the restricted master will be changed into equality for the product $j \in H(u)$.

It is not clear how the generalized upper bound (GUB) branching strategy (see [29], for an application to GAP) can efficiently be applied here. In fact, the use of the GUB branching strategy does not partition the set of feasible solutions into two subsets. Therefore, to cover the whole set of feasible solutions, an important number of branches is needed.

An upper bound at node u is provided by the optimal solution to the master problem (LP_u) or estimated by dualizing some constraints; details on these computations are provided in Section 5.1 and in the Appendix.

4. Heuristics

We have shown (see Proposition 2) that even finding a non-trivial feasible solution to the promotion campaign problem given by the formulation (M1) is NP-hard, which justifies an intensive use of heuristics in practice [5, 6, 12]. In this section, we present five heuristics for the promotion campaign problem. These heuristics are either variants of the algorithms used in practice for application in a bank or specifically developed based on the structure of the problem. The first is a variant of the algorithm developed by FORTIS [10]. It assumes that the variable cost and the profitability of the different clients are identical and that the campaign involves all products. The second heuristic is also based on these simplifying assumptions, but allows for choosing the products to be used for the campaign. The third is a procedure that successively solves a number of *Exact k-item knapsack problems* (E-kKP) (which are kKP's with an equality for the cardinality constraint (32)). It uses an approximation algorithm for solving the (E-kKP) to identify the best product to be offered as well as the selected set of clients at each iteration. The fourth heuristic is also an iterative algorithm, inspired by the Next-Product-To-Buy model used by Knott et al. [12] for an application in a retail bank. Finally, the last heuristic is a depth-first branch-and-price heuristic. In this section, we will say that client i is *active* if it has not yet received M_i offers.

4.1 Heuristic 1

Heuristic 1 is a variant of the algorithm developed by FORTIS. It uses the average cost and the average revenue for each product. These quantities are defined by

$$C_j := \frac{\sum_i c_{ij}}{\text{number of clients}} \quad \text{and} \quad P_j := \frac{\sum_i r_{ij} DFV_{ij}}{\text{number of clients}}.$$

This heuristic ignores the selection of products for the campaign and simply imposes the minimum quantity O_j on the number of offers of each product j. Using a new decision variable $u_j :=$ number of clients that receive an offer for the product j, the simplified formulation used by FORTIS is the following.

$$\begin{aligned}
\text{(M4)} \quad \textsf{maximize} \quad & \sum_{j=1}^{n} \left[(P_j - C_j) u_j - f_j \right] \\
\textsf{subject to} \quad & \sum_{j=1}^{n} \left[P_j - (1+R) C_j \right] u_j \geq (1+R) \left(\sum_{j=1}^{n} f_j \right) \\
& C_j u_j \leq B_j \quad j = 1, \ldots, n, \\
& \\
& O_j \leq u_j \leq m \qquad j = 1, \ldots, n, \\
& u_j \in \mathbb{N} \qquad\quad j = 1, \ldots, n.
\end{aligned}$$

We remark that the problem formulated here is still NP-hard as it includes an integer knapsack problem [11] as a special case. On the other hand, (M4) does not take into account the third constraint (4) of the basic formulation (M1); that is it does not enforce the upper bound constraint on the number of products that can be offered to a client.

Heuristic 1 solves (M4) and uses the outcome to build a solution (potentially infeasible) to our original problem.

2 Heuristic 1

1: for each product j, compute the average revenue P_j and the average cost C_j
2: solve the resulting integer programming formulation (M4)
3: sort products by decreasing profit such that $P_1u_1 \geq P_2u_2 \geq \ldots \geq P_nu_n$
4: for each product j, sort customers in decreasing profit
5: offer the products in increasing order and for each product j offer it to the u_j most profitable active clients
6: choose the best solution between this solution and the trivial solution

4.2 Heuristic 2

Heuristic 2 is an improvement of Heuristic 1; here the average revenue and the average cost per product are still used, but the choice of products to be offered during the campaign is taken into account. Using the same variable u_j defined above and a new binary variable y_j that equals 1 if product j takes part in the campaign and 0 otherwise, the integer programming problem to be solved is

$$
\begin{aligned}
\text{(M5)} \quad \text{maximize} \quad & \sum_{j=1}^{n} \left[(P_j - C_j)u_j - f_j y_j\right] \\
\text{subject to} \quad & \sum_{j=1}^{n} \left[(P_j - (1+R)C_j)\, u_j - (1+R) f_j y_j\right] \geq 0 \\
& C_j u_j \leq B_j \qquad\qquad j = 1, \ldots, n, \\
& O_j y_j \leq u_j \leq m y_j \qquad j = 1, \ldots, n, \\
& y_j \in \{0,1\}, \quad u_j \in \mathbb{N} \qquad j = 1, \ldots, n.
\end{aligned}
$$

Notice that (M5) generalizes (M4), since (M4) arises when we set $y_j = 1$ for all j. This formulation also does not take into account the upper bound on the number of products that can be offered to a client. However, unlike formulation (M4), (M5) does incorporate the choice of products to be offered during the campaign. Heuristic 2 proposes a procedure to find a solution (potentially infeasible) to the promotion campaign problem by solving (M5).

3 Heuristic 2

1: for each product j, compute the average revenue P_j and the average cost C_j
2: solve the resulting integer programming formulation (M5)
3: sort selected products by decreasing profit
4: for each selected product j, sort customers in decreasing profit
5: offer the selected products in increasing order and for each product j offer it to the u_j most profitable active clients
6: choose the best solution between this solution and the trivial solution

Notice that the outcome of Heuristic 2 (as well as of Heuristic 1) may be an infeasible solution; the last line (line 6:) merely guarantees that the value of the solution (potentially infeasible) is nonnegative.

4.3 Heuristic 3

This is a successive Exact k-item knapsack problem (E-kKP) procedure. For a given product j, we have the following (E-kKP):

$$\begin{aligned}
\text{(E-kKPj)} \quad \text{maximize} \quad & \sum_{i=1}^{m} \left[p_{ij} - (1+R)c_{ij}\right] x_{ij} - (1+R)f_j \\
\text{subject to} \quad & \sum_{i=1}^{m} c_{ij} x_{ij} \leq B_j \\
& \sum_{i=1}^{m} x_{ij} = O_j \\
& x_{ij} \in \{0,1\} \qquad i = 1, \ldots, m.
\end{aligned}$$

Remark that the LP relaxation of (E-kKPj) has a solution with either 0 or 2 fractional components. The approximation algorithm for (kKP) is used as 2-approximation algorithm for (E-kKPj).

4 Heuristic 3

1: $val := 0, \quad V := \{1, \ldots, m\}$ //objective value
2: for each $j = 1 \ldots, n$, solve E-kKPj
3: **while** there exists a product j such that $val + z_j^E \geq 0$ **do**
4: select product j^* with the highest profit $z_{j^*}^E$
5: $y_{j^*} := 1$
6: $val := val + z_{j^*}^E$
7: **for** $i \in J_{j^*}^E$ **do**
8: $x_{ij^*} := 1$
9: $M_i := M_i - 1$
10: **if** $M_i = 0$ **then**
11: $V := V \setminus \{i\}$
12: **end if**
13: forbid any further offer of the product j^* to client i //by setting c_{ij^*} greater than B_{j^*}
14: **end for**
15: $O_{j^*} := 1$
16: $f_{j^*} := 0$
17: update B_{j^*}
18: for each $j = 1 \ldots, n$ satisfying $O_j \leq |V|$, solve E-kKPj
19: **end while**
20: **if** $val < \max\{z_j^A,\, j = 1, \ldots, n\}$ **then**
21: $val := \max\{z_j^A,\, j = 1, \ldots, n\}$
22: **end if**

Heuristic 3 describes an iterative procedure for finding a solution to the problem using a 2-approximate solution for (E-kKPj). We denote the objective value found by Approximation algorithm for (kKP) for the problem (E-kKPj) by z_j^E and the set of solution components equal to 1 by J_j^E. The notation z_j^A used by Heuristic 3, refers to the notation of Approximation algorithm for (kKP). Heuristic 3 works as follows: it first selects the product j^* with the highest positive $val + z_{j^*}^E$, this product is then offered to the set of clients $J_{j^*}^E$, the problem is updated and the procedure is repeated until no more product can be offered to clients; the quantity val represents the objective function value of the solution obtained by the algorithm at each stage of its execution. Notice that after each iteration, the reconstructed problem is still a promotion campaign problem.

4.4 Heuristic 4

This heuristic is inspired by the Next-Product-To-Buy model proposed by Knott et al. [12] for an application in a retail bank. For a given product j, the offer is made to active clients in decreasing order of the probabilities r_{ij} until no more budget is left. We refer to this heuristic as Heuristic 4.

5 Heuristic 4

```
1: for j = 1, ..., n do
2:    sort the probabilities r_ij in decreasing order {1, ..., m}
3:    for i = 1, ..., m do
4:       if i is active then
5:          if enough budget then
6:             x_ij* := 1,    M_i := M_i − 1
7:             if M_i = 0 then
8:                i becomes inactive
9:             end if
10:         else
11:            x_ij* := 0
12:         end if
13:      else
14:         x_ij* := 0
15:      end if
16:   end for
17: end for
18: choose the best solution between this solution and the trivial solution
```

Like Heuristic 1 (and Heuristic 2), the last line Heuristic 4 simply guarantee that the value of the solution (potentially infeasible) is nonnegative.

4.5 Heuristic 5

This is a depth-first heuristic based on the branch-and-price logic. This heuristic is used to find a feasible and high-quality solution as quickly as possible. Figure 2 describes the

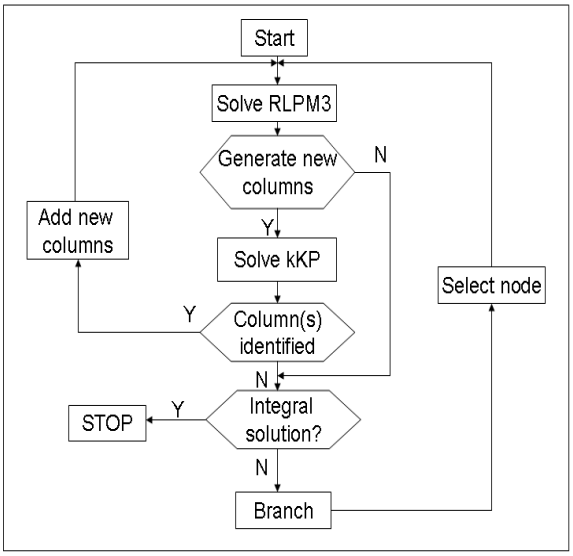

Figure 2: Flow chart of the branch-and-price heuristic (Heuristic 5)

steps to follow. To start the algorithm, we need an initial restricted master problem with a feasible LP relaxation. Therefore, we generate one column (if it is identified by the 2-approximation algorithm) for each product; this is an approximate solution to kKP. We then solve the restricted master problem (RLPM3). Next, columns are generated by running the 2-approximation algorithm for kKP per product built based on the dual variables. If there is no column to add to the master problem RLPM3, we check whether the solution obtained is integral or not. If yes, we stop the algorithm, otherwise we branch and repeat the procedure. On the other hand, if new columns are identified, they are added to the RLPM3 and it is resolved. Remark that here, we do not solve the RLPM3 until optimality. Moreover, as the goal of this heuristic is to find a good feasible solution and not necessarily an optimal one, we do not have to branch such that the solution space is divided evenly. Thus, using the hybrid branching strategy, we choose to branch on the fractional variable x_{ij} closest to 1, and we set $x_{ij} = 1$ in one branch and $x_{ij} = 0$ in the other branch. In case of a tie, we select the variable with the highest revenue p_{ij}. With this branching rule, we are more likely to find a good solution in the node with $x_{ij} = 1$, and we investigate that node first. Similar heuristics have been used successfully for raw materials logistics planning [20] and for the airline crew pairing problem [1].

The first four heuristics present many advantages, among which their simplicity. Their implementation does not require many efforts as the size of the integer program to be solved for Heuristic 1 and Heuristic 2 is usually very small, the solution to the linear program used by Heuristic 3 can be found in linear time and the fourth heuristic is based on an algorithm for sorting, which can easily be done in polynomial time. There is no guarantee, however, that the solutions obtained by these four heuristics are non-zero feasible solutions for the basic model (M1). Heuristic 5, on the other hand, is guaranteed to provide a feasible solution

(often non-zero) but may take more time.

5. Implementation issues

5.1 Branch-and-price algorithm

Initial restricted Master Problem

To start the column generation procedure, an initial restricted master problem with a feasible LP relaxation has to be provided to ensure that the dual variables used to derive the pricing problem are properly computed. We have chosen to start with the solution of Heuristic 3 (which is a non-trivial feasible solution that is usually of good quality and that is obtained quickly) as a starting solution. If Heuristic 3 fails to provide a non-trivial feasible solution, then we generate one column per product corresponding to an optimal solution of the k-item knapsack problem.

Column generation subproblem

Any column with negative reduced cost is a candidate to enter the basis. The exact algorithm (dynamic programming) for the pricing problem (29) finds the column with the lowest reduced cost for each product. If there is a column with negative reduced cost, dynamic programming will always identify a candidate column.

However, solving the LP relaxation (LP_u) at a node u involves the solution of several pricing problems (k-item knapsack problems), which is computationally intensive. As it is not necessary for the column generation scheme to always select the column with the lowest reduced cost, we first use the 2-approximation algorithm for solving the pricing problem. If it fails to identify a column with negative reduced cost, the dynamic programming algorithm is invoked to prove optimality or generate a column with negative reduced cost. This two phase procedure is repeated until an optimal (or near optimal) solution to the LP relaxation is found.

Tailing off and upper bound

At each node of the branching tree, the tailing off effect can be observed. That is: many columns have to be added in order to prove the optimality of the LP solution, but its value (almost) does not change any more [30]. To prevent excessive running times, we have adopted the following rule: if during 30 iterations the objective value has not changed, we stop. Hence, we do not necessarily solve the linear program to optimality. This figure 30 was chosen after many trials. In this way, however, the value found does not provide an upper bound as it is not necessarily optimal. We next describe how we still obtain an upper bound in this case; the method used has been investigated in [15, 30]. At a given node u, consider the LP relaxation problem to be solved (LP_u). Let $\Gamma_j^* = \sum_{i=1}^{m} w_{ij} x_i^*$ be the optimal objective value of the k-item knapsack problem (kKP) (30)-(33) for a given product j, obtained by the dynamic programming algorithm and $J = \{j \in \{1, \ldots, n\} \setminus H(u) : \Gamma_j^* + f_j(1 - (1 + R)d) + v_j < 0\}$.

We have an upper bound given by the right hand side of

$$z \leq \sum_{i=1}^{m} M_i u_i + \sum_{j=1}^{n} v_j - \sum_{j \in H(u) \cup J} \Gamma_j^* + f_j(1 - (1+R)d) + v_j.$$

We detail the procedure to obtain this upper bound in the Appendix. Notice that the computation of this upper bound needs an optimal solution of (kKP). To meet this requirement, we solve the (kKP) problem exactly using the dynamic programming algorithm at the last iteration (30^{th} iteration) of the tailing off.

Branching scheme

We have shown that the hybrid branching strategy is compatible with the pricing problem (Proposition 4). Moreover, in Section 3.3, we have shown that variable dichotomy is compatible with the pricing problem. Therefore, we branch on the fractional variable x_{ij} closest to $\frac{1}{2}$. We set $x_{ij} = 1$ in one branch and $x_{ij} = 0$ in the other branch. In case of a tie, we select the variable with the highest revenue p_{ij}.

5.2 Heuristics

Heuristic 1 & 2

The implementation of the first two heuristics is straightforward. For Heuristic 1 (respectively Heuristic 2), the reduced integer program (M4) (respectively (M5)) is solved using CPLEX 10.2 and the solution then serves as the basis for the implementation of the heuristic.

Heuristic 3

The efficiency of the implementation of Heuristic 3 is essentially based on the efficiency of the algorithm used for solving the linear relaxation of the Exact k-item Knapsack problem. The linear time complexity is achieved using the method proposed by Megiddo [22, 23].

Heuristic 4

Heuristic 4 is based on sorting the r_{ij}'s, which can easily be done in polynomial time.

Heuristic 5

The implementation of Heuristic 5 follows that of the branch-and-price algorithm under the modifications described above. Firstly, we do not use the solution of Heuristic 3 as starting solution, but we generate one column per product (if it is identified by the 2-approximation algorithm). This column is the solution found by the 2-approximation algorithm for (kKP). If for any product there is no column obtained, then we call the dynamic programming algorithm for (kKP). Secondly, at each node we use only the first phase of the two phase procedure of the branch-and-price algorithm. Hence, only the 2-approximation algorithm for (kKP) is used to identify the columns that can be added to the master problem. Therefore, it

may happen that although a column with a negative reduced cost exists for a given product, the 2-approximation algorithm for (kKP) does not identify it. This implies that we do not necessarily compute an optimal solution to the LP relaxation, hence we do not have an upper bound for the best solution reachable from that node. Unlike branch-and-price, however, we do not need an upper bound here. Thirdly, the branching is performed in a greedy fashion, as explained in Section 4.5.

6. Computational experiments

6.1 Generating test instances

The instances used for the experiments are randomly generated, with cost c_{ij} randomly generated from the set $\{1, 2, 3\}$ and the return to the firm $p_{ij} = r_{ij}DFV_{ij}$ is an integer randomly generated between 0 and 16. The choice of these figures is guided by examining the real-life data used by Cohen [5]. The corporate hurdle rate R belongs to the set $\{5\%, 10\%, 15\%\}$. There are six different values for the number of clients m: these are 100, 200 and 300 clients (small size; S1, S2 and S3), 1 000 and 2 000 clients (medium size; M1 and M2) and 10 000 clients for instances of large size (L). For each number of clients, we have three different numbers of products n; these are 5, 10 and 15 products. In total, we have 54 groups of instances, as described in Table 1.

Group	rate R	number of clients (m)	number of products (n)
S1	5%, 10% or 15%	100	5, 10 or 15
S2	5%, 10% or 15%	200	5, 10 or 15
S3	5%, 10% or 15%	300	5, 10 or 15
M1	5%, 10% or 15%	1 000	5, 10 or 15
M2	5%, 10% or 15%	2 000	5, 10 or 15
L	5%, 10% or 15%	10 000	5, 10 or 15

Table 1: Size of the generated inputs

For each group, we generate a minimum quantity commitment bound O_j as a random integer selected between $\lceil \frac{\sum_i M_i}{n} \rceil$ and $\lceil \frac{2\sum_i M_i}{n} \rceil$. We consider three values for the budget B_j, namely a random integer chosen between $O_j \frac{\sum_i c_{ij}}{m}$ and $2\frac{\sum_i c_{ij}}{n}$ and the two extreme budgets which are $\lfloor O_j \frac{\sum_i c_{ij}}{m} \rfloor$ and $\lceil 2\frac{\sum_i M_i}{n} \frac{\sum_i c_{ij}}{m} \rceil$. The fixed cost f_j is a random integer between $\frac{O_j}{2m(1+R)} \sum_i [p_{ij} - (1+R)c_{ij}]$ and $\frac{O_j}{m(1+R)} \sum_i [p_{ij} - (1+R)c_{ij}]$. These bounds have been chosen in such a way that the instances become feasible and consistent. We generate one instance with a small upper bound M_i for any client, selected between 1 and $\frac{n}{5}$, and another one with a large random upper bound M_i for each client, selected between $\lceil \frac{n}{3} \rceil$ and $\lceil \frac{2n}{3} \rceil$. In conclusion, we have in total $3 \times 2 \times 54 = 324$ test instances.

Since Heuristic 4 requires as input r_{ij} (and not p_{ij}), we have decided to generate for each of the 324 test instances, three different sets of values for r_{ij}. The first set is generated randomly from $]0, 1[$, while the second is also randomly generated in $]0, 1[$ but proportional to c_{ij} and the last is inversely proportional to p_{ij}. Note that by choosing r_{ij} in $]0, 1[$, we discard certainty. These $3 \times 324 = 972$ test instances are solved using Heuristic 4.

We have implemented all algorithms in Visual Studio C++ 2005; all the experiments were run on a Dell Optiplex GX620 personal computer with Pentium R processor with 2.8 GHz clock speed and 1.49 GB RAM, equipped with Windows XP. CPLEX 10.2 was called as subroutine for solving the linear programs.

6.2 Computational results

In this section, computation time is expressed in seconds, while Gap is a percentage $\frac{|z_{AP}-z_{UB}|}{z_{UB}} \times 100\%$, where z_{UB} is either the optimal objective value or an upper bound of the optimal objective value and z_{AP} is the objective value obtained by our algorithm.

			Basic formulation				Set covering formulation			
Group	n	M	Gap	Time			Gap	Time		
				min	mean	max		min	mean	max
S1	5	s	8.33	0.06	0.16	0.31	5.05	4.88	16.13	51.70
		ℓ	6.51	0.08	0.30	0.55	0.03	1.70	95.62	784.97
	10	s	3.11	0.05	0.13	0.23	0.62	6.09	13.71	24.98
		ℓ	3.26	0.08	0.26	0.50	0.18	3.70	14.31	41.33
	15	s	2.96	0.05	0.14	0.27	0.45	6.89	11.59	34.56
		ℓ	1.28	0.06	0.25	0.53	0.07	1.73	105.06	805.14
S2	5	s	2.51	0.11	0.31	0.67	0.78	166.03	705.97	2862.80
		ℓ	12.80	0.19	0.93	2.31	5.47	0.47	115.57	531.11
	10	s	3.37	0.11	0.46	0.91	0.21	105.60	287.50	518.71
		ℓ	10.51	0.20	1.06	2.28	0.80	0.45	305.50	1047.98
	15	s	2.47	0.11	0.43	0.83	0.15	80.75	190.46	271.88
		ℓ	5.00	0.16	1.12	2.23	0.00	120.67	225.42	458.99
S3	5	s	8.65	0.17	0.97	1.99	7.57	784.69	3004.44	11052.67
		ℓ	5.92	0.44	2.14	4.28	0.20	14.63	674.21	1575.14
	10	s	3.48	0.19	0.51	0.88	0.58	1407.73	2322.51	3404.60
		ℓ	16.73	0.47	1.91	3.86	0.03	0.91	1838.27	8277.15
	15	s	2.23	0.19	0.42	0.75	0.69	1766.64	3158.25	4804.45
		ℓ	13.65	0.48	2.54	5.74	0.77	1.09	588.26	1480.95

Table 2: LP relaxation of the basic formulation and the set covering formulation

Table 2 shows a comparison of the LP relaxation of the basic formulation and the set covering formulation. Notice that each cell is the average of 9 values, described by three values for R and three values for the budget B. The table confirms the theoretical result obtained in Section 3.1 that the set covering formulation is strong: that is, it gives a solution very close to the optimal solution compared to the solution obtained by the basic formulation. This difference is reflected by Gap for the set covering formulation. We observe that Gap is less than 1% for many groups. However, this quality of the LP relaxation of the set covering formulation comes with a relatively high computation time. Notice that this computation time increases with the size of the problem (number of clients). Table 2 explicitly displays the trade-off between the solution quality and the computation time. Remark that the computation time displayed in Table 2 for the set covering formulation is not necessarily the computation time spent by the branch-and-price algorithm at the root node as we stop

solving the root node when we go through 30 iterations without an improvement of the objective value (as discussed in Section 5.1).

Table 3 compares three different strategies of traversing the branching tree. These are breadth first, depth first, and best first. The main difference between these three strategies is observed after the branching, when selecting the child to investigate first [29].

			breadth first		depth first		best first	
	n	M	Time	Nr Nodes	Time	Nr Nodes	Time	Nr Nodes
S1	5	s	10131.57	178	7365.78	74	2031.13	26
		ℓ	64.68	8	38.75	6	72.85	2
	10	s	5057.54	936	1141.08	210	3035.86	410
		ℓ	9967.78	754	20172.64	1395	8587.53	208

Table 3: Comparison of different tree traversal strategies for the branch-and-price algorithm

The comparison is made for 36 instances from Group S1. We clearly see from Table 3 that the output of the breadth-first branch-and-price algorithm is dominated either by the output of the depth-first branch-and-price algorithm or by the output of the best-first branch-and-price algorithm. Remark that the number of nodes might be important here as more nodes is likely to imply more columns generated. The generation of these columns is time consuming, and too many columns may sometimes lead to a memory problem for our branch-and-price algorithm.

Table 4 displays the results of the exact algorithms for the promotions campaigns problem. The basic formulation (M2) is solved using the MIP (mixed-integer programming) solver of CPLEX 10. In Table 4, the minimum, the mean (average) and the maximum computation time are recorded. For Group S1, CPLEX was able to solve each instance. However, the computation time increases when we move to Group S2. For that group, CPLEX could not solve four instances, due to memory problems. Amongst these four instances, three have a small upper bound per client M_i (more precisely $M_i = 1$) and only one was an instance with large upper bound. For Group S3, up to ten instances were not solved by CPLEX. In that group, all unsolved instances have a small upper bound M_i. The above observations coupled with an analysis of the computation time given in Table 4 for the basic formulation lead to the following conclusions. Computation time seems to increase exponentially with the size of the instance (number of clients). Moreover, instances with small upper bound M_i seem to be harder to solve than instances with large upper bound.

Table 4 also shows the computation times obtained by the branch-and-price algorithms. We concentrate only on depth-first and best-first branch-and-price algorithms as Table 3 has shown that these two dominate the breadth-first branch-and-price algorithm. It is clear that using the Mixed Integer Optimizer of CPLEX 10.2 for solving the basic formulation (M2) (enhanced with the default cutting planes) is much faster than the branch-and-price approaches. Although the pricing problem is solved fast and Gap is smaller at the root node, the number of nodes in the branching tree can be excessive, as witnessed in Table 4. We do not apply the branch-and-price algorithms for solving the instances in Group S3.

Group	n	M	Basic formulation			depth first			best first		
			Time			Time			Time		
			min	mean	max	min	mean	max	min	mean	max
S1	5	s	1.72	57.39	190.13	103.69	7365.78	21135.56	279.97	2031.13	4999.85
		ℓ	0.11	3.91	12.28	11.95	38.75	75.78	11.20	72.85	168.86
	10	s	1.33	27.90	114.84	24.98	1141.08	11052.67	578.39	3035.86	5493.34
		ℓ	0.17	6.05	35.95	61.72	20172.64	35798.08	74.16	8587.53	16683.14
	15	s	1.50	22.57	74.73	17259.00	39845.535	62432.07*	3038.83	19521.5	36004.17*
		ℓ	0.11	3.22	14.03	23.81	31117.1	62210.36	23.56	18014.18	36004.79
S2	5	s	0.23	253.47	1036.68	2862.80	36725.30	62143.55	2862.80	36382.61	51143.56*
		ℓ	0.44	138.18	1094.44*	358.72	12125.45	36017.63	251.53	15699.19	36101.53
	10	s	3.58	2246.54	16249.26*	518.71	36138.39	57844.56	422.7	36696.35	57842.13*
		ℓ	0.33	20.40	124.16	336.17	12010.04	36021.12	631.33	18334.06	36036.80
	15	s	3.59	1811.47	14259.19*	1575.14	12545.51	36061.38	444.75	36209.22	64422.08*
		ℓ	0.30	9.47	53.83	458.99	5742.13	11052.67	179.95	18183.22	36186.49
S3	5	s	0.48	241.79	1311.32*	×	×	×	×	×	×
		ℓ	2.63	103.94	596.19	×	×	×	×	×	×
	10	s	12.81	823.01	3213.22*	×	×	×	×	×	×
		ℓ	0.75	809.26	5646.58	×	×	×	×	×	×
	15	s	1.53	127.03	370.22*	×	×	×	×	×	×
		ℓ	0.66	172.78	1490.54	×	×	×	×	×	×

* identifies the groups where some instances could not be solved.

Table 4: Basic formulation and Branch-and-price algorithms

		Heuristic 2				Heuristic 3				Heuristic 5			
Group	n	Pos	Gap	Time	Feas	Pos	Gap	Time	Feas	Pos	Gap	Time	Feas
S1	5	61.11	84.99	0.01	1.64	100	7.07	0.14	100	100	2.90	95.34	100
	10	66.67	83.02	0.01	1.50	100	10.27	0.14	100	100	6.68	168.14	100
	15	77.78	83.42	0.02	0.00	100	11.51	0.15	100	100	6.43	363.33	100
S2	5	50.00	86.85	0.02	0.00	100	8.19	0.55	100	100	14.81	2190.56	100
	10	77.78	87.37	0.01	2.57	100	9.22	0.62	100	100	6.77	1892.39	100
	15	58.82	83.62	0.02	1.70	100	9.22	0.62	100	100	14.72	2217.17	100
S3	5	41.18	82.97	0.02	0.00	100	8.46	2.93	100	58.82*	4.68	869.59	100
	10	41.18	87.55	0.01	0.00	100	8.70	9.97	100	58.82*	14.15	339.68	100
	15	41.18	82.65	0.01	0.00	100	9.47	1.74	100	58.82*	13.12	434.03	100

* identifies the groups where there is an effect of the time limit.

Table 5: Heuristics comparison

In Table 5, the outcomes of the heuristics are exhibited. Again, Gap is a percentage $\frac{|z_{AP}-z_{UB}|}{z_{UB}} \times 100\%$, where z_{UB} is the optimal objective value and z_{AP} is the objective value obtained by the heuristic. Pos is the percentage of (potentially infeasible) solutions with positive objective value and Feas is the percentage of feasible solutions among solutions with positive objective value. Unlike the previous tables, here each cell is the average of 18 values, described by three values for R, three values for the budget B and the two values of M. The experimental results of Heuristic 1 are not displayed in Table 5 because the solutions produced by Heuristic 2 are better.

The results of Table 5 confirm the existence of a trade-off between computation time and solution quality. Heuristic 2 is the fastest heuristic in the table. This is mainly because the reduced formulation to be solved is very small (the size is proportional to the number of products). This low computation time comes with a high Gap (more than 80%) which expresses a very poor solution quality. Moreover, the solutions obtained are usually not feasible (at most 3% of the solutions with positive objective value). Heuristic 3, however, gives a feasible solution with positive objective value for all the test instances. Moreover, Gap (at most 12%) and the computation time (at most 10s) make it an attractive heuristic. The solutions provided by Heuristic 5 are of good quality with Gap of at most 15%. However, unlike Heuristic 3, Heuristic 5 requires much more time. This computation time increases with the size of the problem. Therefore, Heuristic 5 is applied for Group S3 (instances with 300 clients) with a time limit of one hour; we were able to provide solutions with positive objective value (and feasible) for more than 50% of these instances.

Table 6 depicts the results of Heuristic 4 for the three categories of probabilities (random, proportional to cost and inversely proportional to revenue). Although Heuristic 4 does not find any feasible solution, we see different results for the three categories. The best behavior of Heuristic 4 is observed for the random probabilities (which is close to the real life situation) with the percentage of solutions with positive objective value around 80% and a gap usually less than 80%. Also, remark that Heuristic 4 achieves the smallest running time among all heuristics.

Heuristic 4													
		Cost				Revenue				Random			
Group	n	Pos	Gap	Time	Feas	Pos	Gap	Time	Feas	Pos	Gap	Time	Feas
S1	5	5.56	88.61	0.00	0.00	16.67	94.48	0.00	0.00	83.33	80.73	0.00	0.00
	10	11.11	82.10	0.00	0.00	55.56	88.89	0.00	0.00	88.89	75.26	0.00	0.00
	15	11.11	80.22	0.00	0.00	72.22	89.35	0.00	0.00	94.44	78.35	0.00	0.00
S2	5	6.25	89.01	0.00	0.00	25.00	91.91	0.00	0.00	62.50	80.98	0.00	0.00
	10	33.33	84.52	0.00	0.00	38.89	83.67	0.00	0.00	94.44	81.20	0.00	0.00
	15	23.53	84.24	0.00	0.00	47.06	88.49	0.00	0.00	88.24	76.62	0.00	0.00
S3	5	17.65	78.31	0.00	0.00	47.06	86.40	0.00	0.00	82.35	75.36	0.00	0.00
	10	17.65	81.76	0.00	0.00	47.06	90.34	0.00	0.00	82.35	80.64	0.00	0.00
	15	17.65	70.95	0.00	0.00	47.06	84.12	0.00	0.00	82.35	72.71	0.00	0.00

Table 6: Heuristic 4

To summarize the comparison between the five heuristics, we formulate the following advice. If the solution quality is the only criterion to be taken into account, the use of Heuristic 5 or Heuristic 3 is advised if an efficient exact algorithm is not available. However, if both the computation time and the solution quality are relevant, we strongly recommend the use of Heuristic 3. Finally, we advise not to use Heuristic 2 or Heuristic 4 unless no other choice is available as they do not guarantee feasibility nor good quality of the solutions.

6.3 Large instances

		Heuristic 2				Heuristic 3				Heuristic 4 (random)			
Group	n	Pos	Gap	Time	Feas	Pos	Gap	Time	Feas	Pos	Gap	Time	Feas
M1	5	38.89	84.51	0.01	0.00	100	12.56	156.26	100	83.33	83.73	0.01	0.00
	10	50.00	88.33	0.01	2.00	100	13.45	454.52	100	88.89	84.41	0.01	0.00
	15	55.56	82.41	0.01	0.00	100	14.76	324.82	100	100	78.06	0.01	0.00
M2	5	44.44	83.36	0.01	2.25	100	14.01	2135.32	100	66.67	85.05	0.01	0.00
	10	44.44	86.44	0.01	2.25	100	13.35	2331.94	100	94.44	82.20	0.01	0.00
	15	61.11	88.10	0.01	0.00	100	12.82	1197.70	100	88.89	81.67	0.01	0.00
L	5	38.89	90.70	0.03	0.00	100	34.04	3573.92	100	100	84.74	0.08	0.00
	10	38.89	78.25	0.04	0.00	100	33.52	3143.41	100	100	78.91	0.08	0.00
	15	38.89	85.45	0.04	1.50	100	37.80	3353.81	100	100	75.40	0.07	0.00

Table 7: Heuristics comparison for large instances

In Table 7, Pos and Feas have the same meaning as in Table 5, but Gap is a percentage $\frac{|z_{AP}-z_{UB}|}{z_{UB}} \times 100\%$, where z_{UB} is an upper bound of the the optimal objective value obtained by solving the LP relaxation and z_{AP} is the objective value obtained by the heuristic.

Due to time constraints, Heuristic 3 was run for at most one hour per instance in Group L. Hence, the solution output by the Heuristic 3 for instances in that group is not necessarily the best solution it can provide. Similarly, for some instances, the LP relaxation was not solved until optimality, and we impose a time limit of one hour and consider the value obtained at that time as an upper bound of the optimal objective value.

Table 7 confirms the difficulty experienced by Heuristic 2 and Heuristic 4 in finding a feasible solution. Moreover, we observe a slight reduction (respectively increase) in the percentage of non-zero solutions for Heuristic 2 (respectively Heuristic 4) compared with smaller instances, while the gap remains quite large (more than 80% for Heuristic 2 and more than 75% for Heuristic 4) and the percentage of feasible solutions is not promising. The only good news comes from the computation time, which increases very slowly.

Heuristic 3, by contrast, behaves very well for large instances: Gap is less than 15% for Group M1 and Group M2. Although this Gap increases up to 38% for the instances with 10 000 clients, Heuristic 3 still outperforms all the other heuristics.

In practice, there may be instances with millions of clients [5]. When dealing with such instances, one often-used technique is to cluster the clients into groups (these techniques are also called *aggregate techniques* [5]). Next, clients within a group are treated as being identical. We point out here that a variant of (M1) provides a model for this situation where x_{ij} is then defined as the number of clients of group i that receive the product j and the parameters c_{ij} and p_{ij} are redefined accordingly. Let T be the number of groups thus built and $size(i)$ be the cardinality of group i.

$$
\begin{aligned}
\text{(GM1)} \quad \text{maximize} \quad & \sum_{i=1}^{T}\sum_{j=1}^{n}(p_{ij}-c_{ij})\,x_{ij} - \sum_{j=1}^{n} f_j y_j \\
\text{subject to} \quad & \sum_{i=1}^{T}\sum_{j=1}^{n} p_{ij}x_{ij} \geq (1+R)\left[\sum_{i=1}^{T}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j y_j\right] \\
& \sum_{i=1}^{T} c_{ij}x_{ij} \leq B_j \qquad j=1,\ldots,n, \\
& \sum_{j=1}^{n} x_{ij} \leq size(i)M_i \qquad i=1,\ldots,T, \\
& \sum_{i=1}^{T} x_{ij} \leq m y_j \qquad j=1,\ldots,n, \\
& \sum_{i=1}^{T} x_{ij} \geq O_j y_j \qquad j=1,\ldots,n, \\
& y_j \in \{0,1\} \qquad x_{ij} \in \{0,\ldots,size(i)\} \qquad i=1\ldots,T,\ j=1,\ldots,n.
\end{aligned}
$$

Often in practice, instead of solving the integer program (GM1), its LP relaxation is solved and the outcome is used to build an assignment problem for each group.

7. Extensions

In this section we deal with two additional constraints regularly encountered in financial institutions [5, 12]. The first constraint is a single time window constraint. So far, our formulations take into account only the probability that client i reacts positively to an offer of product j, missing the question of *when* client i is likely to accept an offer of j. This

condition can be incorporated in our formulations by taking for a given time window T the probability r_{ij}^T that client i reacts positively to an offer of product j over the next period T. These quantities are computed using well known statistical and probability tools [12]. The extension with sequentially ordered products over multiple time windows [16, 25] is an avenue for future work.

The second constraint is related to the maximum number of offers that can be made for a given product during the campaign. This constraint is easily incorporated into the basic formulation by adding a constraint of the form $\sum_i x_{ij} \leq N_j y_j$ for each product j, where N_j represents the maximum number of offers allowed for the product j. The set covering formulation (M3) remains valid under a slight modification in the definition of S_{pj} such that it contains at least O_j clients and at most N_j clients. The major effect appears in the pricing problem: instead of solving the k-item knapsack problem (kKP) ((30)-(33)), we have to solve the following $k_1|k_2$-item knapsack problem:

$$(k_1|k_2\text{KP}) \quad \text{minimize} \quad \sum_{i=1}^{m} w_i x_i \tag{43}$$

$$\text{subject to} \quad \sum_{i=1}^{m} c_i x_i \leq B \tag{44}$$

$$k_1 \leq \sum_{i=1}^{m} x_i \leq k_2 \tag{45}$$

$$x_i \in \{0,1\} \qquad i = 1, \ldots, m. \tag{46}$$

The dynamic programming recursion applied for the k-item knapsack problem is also valid for the $k_1|k_2$-item knapsack problem if we replace k by $\min\{k, k_2\}$, while the 2-approximation algorithm is valid without any modification. This is due to the fact that a solution to the $k_1|k_2$-item knapsack problem has at most two fractional components. This second constraint is also easily incorporated in all the heuristics defined above after some minor modifications.

8. Conclusions and future works

This text explicitly models the promotion campaign problem taking into account both business constraints and customer preferences and specificities. Our research shows that the problem is strongly NP-hard and that it is unlikely that a constant factor approximation algorithm can be proposed for solving this problem. We have also presented a set covering formulation for the promotion campaign problem in which each product is associated with a subset of clients (which can be empty) in the optimal solution and developed a branch-and-price algorithm for solving it. We have shown that this last formulation is stronger than the basic formulation. These two formulations are used to produce optimal solutions to the promotion campaign problem. Experimental results confirm that these two formulations are limited by the size of instances that they can efficiently solve, which makes their application more suited for business-to-business applications in which variable (and fixed) costs are more important and the number of clients is not very large (around 1000 clients [9]).

To extend the application to a business-to-consumer environment with considerably more customers, we have presented five heuristics. Some of these heuristics are currently used

in practice (Heuristic 1, 2, 4), while others are new (Heuristic 3, 5). Based on extensive experimental results, we provide a comparison and comments on the efficiency and quality of the results obtained using the different formulations and the heuristics. These results clearly show a trade-off between computation time and solution quality and suggest the use of optimal algorithms for small and medium size instances, while heuristics are prererable for large size instances and when time is an important factor.

An important immediate further research direction that might be pursued is an extension of this work to multichannel offers where the products can be offered through many different channels and each channel has its own constraints (e.g. minimum and maximum number of offers through this channel [5]). For the long term, we may extend this work to take into account sequentially ordered products over time that is when different related products are offered over multiple time windows [16, 25].

References

[1] C. Barnhart, E. Johnson, and M. Savelsbergh. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[2] D. Bertsimas and J.N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, 1997.

[3] R. C. Bruner, K. M. Eades, R. S. Harris, and R. C. Higgins. Best practices in estimating the cost of capital: survey and synthesis. *Financial Practice and Education*, 8:13–28, 1998.

[4] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research*, 123:333–345, 2000.

[5] M. Cohen. Exploiting response models - optimizing cross-sell and up-sell opportunities in banking. *Information Systems*, 29:327–341, 2004.

[6] B. De Reyck and Z. Degraeve. Broadcast scheduling for mobile advertising. *Operations Research*, 51:509–517, 2003.

[7] F.R. Dwyer. Customer lifetime valuation to support marketing decision making. *Journal of Direct Marketing*, 11:6–13, 1997.

[8] M.R. Garey and D.S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Co., 1979.

[9] Air Transport Action Group. The economic and social benefits of air transport. Technical report, Air Transport Action Group, December 2007.

[10] E. Hellinckx. Customer relationship management: De optimalisatie van de planning van campagnes. Master's thesis, KULeuven, 2004 (in Dutch).

[11] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer, 2004.

[12] A. Knott, A. Hayes, and S.A. Neslin. Next-product-to-buy models for cross-selling applications. *Journal of Interactive Marketing*, 16:59–75, 2002.

[13] P. Kotler and G. Armstrong. *Principles of marketing*. Pearson Prentice Hall, 2006.

[14] V. Kumar, G. Ramani, and T. Bohling. Customer lifetime value approaches and best practice applications. *Journal of Interactive Marketing*, 18:60–72, 2004.

[15] L.S. Lasdon. *Optimization theory for large systems*. The Macmillan Company, 1970.

[16] S. Li, B. Sun, and R.T. Wilcox. Cross-selling sequentially ordered products: An application to consumer banking services. *Journal of Marketing Research*, XLII:233–239, 2005.

[17] A. Lim, F. Wang, and Z. Xu. On the selection and assignment with minimum quantity commitments. *Lecture Notes in Computer Science*, 3106:102–111, 2004.

[18] A. Lim, F. Wang, and Z. Xu. A transportation problem with minimum quantity commitment. *Transportation Science*, 40:117–129, 2006.

[19] A. Lim and Z. Xu. The bottleneck problem with minimum quantity commitment. *Naval Research Logistics*, 53:91–100, 2006.

[20] Z. Luo, L. Tang, and W. Zhang. Using branch-and-price algorithm to solve raw materials logistics planning problem in iron and steel industry. *2007 International Conference on Management Science and Engineering*, pages 529–536, 2007.

[21] S. Martello and P. Toth. *Knapsack problems: Algorithms and computer implementation*. Jown Wiley and Sons, 1990.

[22] N. Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the Association for Computing Machinery*, 31:114–127, 1984.

[23] N. Megiddo and A. Tamir. Linear time algorithms for some separable quadratic programming problems. *Operations Research Letters*, 13:203–211, 1993.

[24] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*. Wiley, 1988.

[25] A. Prinzie and D. Van Den Poel. Investigating purchasing-sequence patterns for financial services using Markov, MTD and MTDg models. *European Journal of Operational Research*, 170:710–734, 2006.

[26] A. Prinzie and D. Van Den Poel. Predicting home-appliance acquisition sequences: Markov/Markov for discrimination and survival analysis for modeling sequencial information in NPTB models. *Decision Support Systems*, 44:28–45, 2007.

[27] W. Reinartz, J.S. Thomas, and V. Kumar. Balancing acquisition and retention resources to maximize customer profitability. *Journal of Marketing*, 69:63–79, 2005.

[28] L. Ryals. Making customer relationship management work: The measurement and the profitable management of customer relationships. *Journal of Marketing*, 69:252–261, 2005.

[29] M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45:831–841, 1997.

[30] F. Vanderbeck and L.A. Wolsey. An exact algorithm for IP column generation. *Operations Research Letters*, 19:151–159, 1996.

[31] L.A. Wolsey. *Integer programming*. Wiley, 1998.

Appendix: upper bound computation

In this section, we describe how to obtain an upper bound at a node u of a branch-and-price tree where we observe the tailing off effect. The method used has been investigated in [15,30]. At a given node u, the LP relaxation problem to be solved (LP_u) can be reformulated as follows.

$$(\mathrm{LP}_u) \quad \mathsf{max} \quad \sum_{j=1}^{n} \left[\sum_{p=1}^{k_j} \left(\sum_{i \in S_{pj}} (p_{ij} - c_{ij}) - f_j \right) z_{pj} \right] \ (= z) \tag{47}$$

$$\mathsf{subject\ to} \quad \sum_{j=1}^{n} \left[\sum_{p=1}^{k_j} \left(\sum_{i \in S_{pj}} (p_{ij} - (1+R)c_{ij}) - (1+R)f_j \right) z_{pj} \right] \geq 0 \ (d \leq 0) \tag{48}$$

$$\sum_{j=1}^{n} \sum_{p: i \in S_{pj}} z_{pj} \leq M_i \qquad (u_i \geq 0) \qquad i = 1, \ldots, m, \tag{49}$$

$$\sum_{p: R_j^u \subseteq S_{pj}} z_{pj} = 1 \qquad (v_j \text{ free}) \qquad j \in H(u), \tag{50}$$

$$\sum_{p=1}^{k_j} z_{pj} \leq 1 \qquad (v_j \geq 0) \qquad j \in \{1, \ldots, n\} \setminus H(u), \tag{51}$$

$$z_{pj} \in \mathbb{E}_j^u, \quad j = 1, \ldots, n, \tag{52}$$

where $\mathbb{E}_j^u$ is defined by

$$\begin{aligned}
&\text{if } j \notin H(u) \cup L(u), && \mathbb{E}_j^u = \{z_{pj} \geq 0 : \ p \in \{1, \ldots, k_j\}\},\\
&\text{if } j \in H(u) \setminus L(u), && \mathbb{E}_j^u = \{z_{pj} \geq 0 : \ p \in \{1, \ldots, k_j\}, \ R_j^u \subseteq S_{pj}\},\\
&\text{if } j \in L(u) \setminus H(u), && \mathbb{E}_j^u = \{z_{pj} \geq 0 : \ p \in \{1, \ldots, k_j\}\}, \ N_j^u \cap S_{pj} = \emptyset\},\\
&\text{if } j \in H(u) \cap L(u), && \mathbb{E}_j^u = \{z_{pj} \geq 0 : \ p \in \{1, \ldots, k_j\}\}, \ R_j^u \subseteq S_{pj}, \ N_j^u \cap S_{pj} = \emptyset\}.
\end{aligned}$$

By dualizing the constraints (48)-(51), we obtain:

$$
\begin{aligned}
z \;\leq\; \mathsf{max} \quad & \sum_{j=1}^{n}\left[\sum_{p}\left(\sum_{i\in S_{pj}}(p_{ij}-c_{ij})-f_j\right)z_{pj}\right] \\
& -d\left\{\sum_{j=1}^{n}\left[\sum_{p}\left(\sum_{i\in S_{pj}}(p_{ij}-(1+R)c_{ij})-(1+R)f_j\right)z_{pj}\right]\right\} \\
& +\sum_{i=1}^{m}M_iu_i-\sum_{j=1}^{n}\left[\sum_{p}(\sum_{i\in S_{pj}}u_i)z_{pj}\right]+\sum_{j=1}^{n}v_j-\sum_{j=1}^{n}\sum_{p}v_jz_{pj} \\
\mathsf{subject\ to} \quad & z_{pj}\in\mathbb{E}_j^u, \quad j=1,\ldots,n.
\end{aligned}
$$

The above inequality is rewritten as:

$$
\begin{aligned}
z-\sum_{i=1}^{m}M_iu_i-\sum_{j=1}^{n}v_j \;\leq\; \mathsf{max} \quad & \sum_{j=1}^{n}\left[\sum_{p}-\left\{\left(\sum_{i\in S_{pj}}p_{ij}(1-d)+c_{ij}(1-(1+R))+u_i\right)\right.\right. \\
& \qquad \left.\left.+f_j(1-(1+R)d)+v_j\right\}z_{pj}\right] \\
\mathsf{subject\ to} \quad & z_{pj}\in\mathbb{E}_j^u, \quad j=1,\ldots,n \\
\leq \quad \mathsf{max} \quad & \sum_{j=1}^{n}\left[\sum_{p}-\left\{\left(\sum_{i\in S_{pj}}w_{ij}\right)+f_j(1-(1+R)d)+v_j\right\}z_{pj}\right] \\
\mathsf{subject\ to} \quad & z_{pj}\in\mathbb{E}_j^u, \quad j=1,\ldots,n \\
\leq \quad \mathsf{max} \quad & \sum_{j=1}^{n}\left[\sum_{p}-\left\{\left(\begin{array}{c}\mathsf{min}\\ x_i\in\mathbb{X}_j^u\end{array}\sum_{i=1}^{m}w_{ij}x_i\right)+f_j(1-(1+R)d)+v_j\right\}z_{pj}\right] \\
\mathsf{subject\ to} \quad & z_{pj}\in\mathbb{E}_j^u, \quad j=1,\ldots,n
\end{aligned}
$$

where $w_{ij}=p_{ij}(d-1)+c_{ij}(1-(1+R)d)+u_i$ and $\mathbb{X}_j^u$ is defined by

$$
\begin{aligned}
&\text{if } j\notin H(u)\cup L(u), \qquad \mathbb{X}_j^u=\left\{x_i\in\{0,1\}:\ \sum_{i=1}^{m}c_{ij}x_i\leq B_j,\ \sum_{i=1}^{m}x_i\geq O_j\right\},\\
&\text{if } j\in H(u)\setminus L(u), \qquad \mathbb{X}_j^u=\left\{x_i\in\{0,1\}:\ \sum_{i=1}^{m}c_{ij}x_i\leq B_j,\ \sum_{i=1}^{m}x_i\geq O_j,\ x_i=1,\ \forall i\in R_j^u\right\},\\
&\text{if } j\in L(u)\setminus H(u), \qquad \mathbb{X}_j^u=\left\{x_i\in\{0,1\}:\ \sum_{i=1}^{m}c_{ij}x_i\leq B_j,\ \sum_{i=1}^{m}x_i\geq O_j,\ x_i=0,\ \forall i\in N_j^u\right\},\\
&\text{if } j\in H(u)\cap L(u), \qquad \mathbb{X}_j^u=\left\{x_i\in\{0,1\}:\ \sum_{i=1}^{m}c_{ij}x_i\leq B_j,\ \sum_{i=1}^{m}x_i\geq O_j,\ \begin{array}{c}x_i=1,\ \forall i\in R_j^u,\\ x_i=0,\ \forall i\in N_j^u\end{array}\right\}.
\end{aligned}
$$

Let $\Gamma_j^* = \sum_{i=1}^m w_{ij} x_i^*$ be the optimal objective value of the k-item knapsack problem (kKP) (30)-(33) obtained by the dynamic programming algorithm. We have

$$\begin{aligned} z - \sum_{i=1}^{m} M_i u_i - \sum_{j=1}^{n} v_j \quad &\leq \quad \mathsf{max} \quad \sum_{j=1}^{n} \left[\sum_{p} - \left\{ \Gamma_j^* + f_j(1-(1+R)d) + v_j \right\} z_{pj} \right] \\ &\qquad \mathsf{subject\ to} \quad z_{pj} \in \mathbb{E}_j^u, \quad j = 1, \ldots, n \\ &\leq \quad \mathsf{max} \quad \sum_{j=1}^{n} \left[- \left\{ \Gamma_j^* + f_j(1-(1+R)d) + v_j \right\} \sum_{p} z_{pj} \right] \\ &\qquad \mathsf{subject\ to} \quad z_{pj} \in \mathbb{E}_j^u, \quad j = 1, \ldots, n \end{aligned}$$

We know that $\sum_p z_{pj} = 1$ for $j \in H(u)$ and $\sum_p z_{pj} \leq 1$ for $j \in \{1, \ldots, n\} \setminus H(u)$. Let $J = \{j \in \{1, \ldots, n\} \setminus H(u) : \Gamma_j^* + f_j(1-(1+R)d) + v_j < 0\}$, we set $\sum_p z_{pj} = 1$ for $j \in J$ and $\sum_p z_{pj} = 0$ for $j \notin H(u) \cup J$. At a node u, we have an upper bound given by the right hand side of

$$z \leq \sum_{i=1}^{m} M_i u_i + \sum_{j=1}^{n} v_j - \sum_{j \in H(u) \cup J} \Gamma_j^* + f_j(1-(1+R)d) + v_j.$$